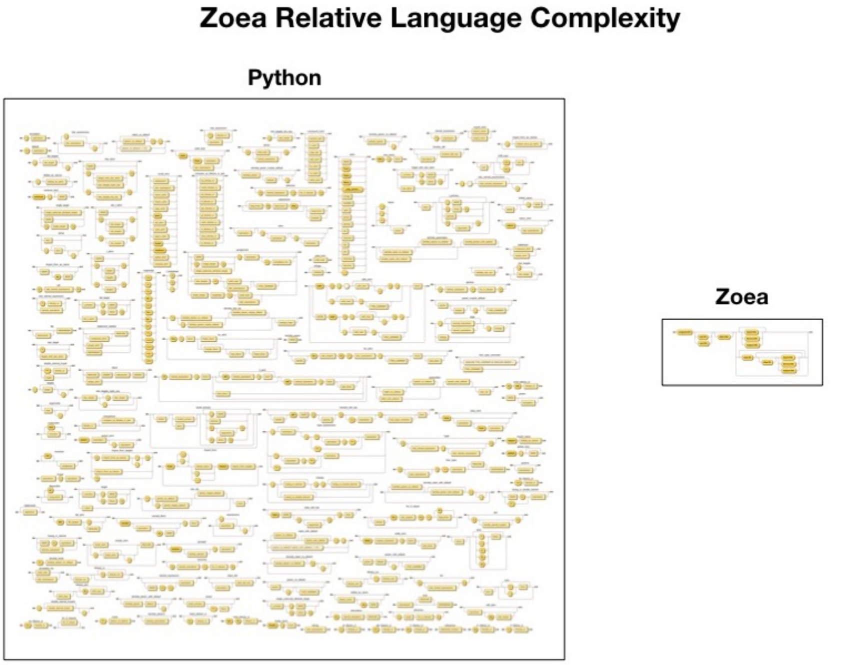
Research Note 15

Where Does Software Complexity Come From?

Edward McDaid & Sarah McDaid

16 Sep 2021

Software complexity is a measure of how difficult it is to produce a piece of code. Does this complexity come from the problem that is being solved or from the programming language that is used to create the solution, and how can we tell?



© zoea.co.uk

Software developers commonly use a number of metrics to describe their code. For example, most coders will talk about the size of a program in terms of the number of characters or lines of code it contains. Lines of code is an approximate metric as it depends on factors such as how the code is formatted which can vary - but it is still widely used.

The complexity of a piece of code is another important< measure. This is intended to capture how difficult software is to write or to understand. There are a number of different complexity measures in use of which cyclomatic complexity is probably the best known. This basically counts how many different paths there are through a given piece of code from the start to the end. Software elements like loops, conditional branches and function calls increase the complexity. The result is a single number that allows different programs - or the same program at different times - to be compared. This allows us to say things like program A is less complex than program B, or that program C is getting more complex over time.

When we produce a program that solves a given problem some of the complexity of the resulting program is certainly due to the problem itself. Complex problems will often require complex code solutions. What might be surprising is that some of the complexity also comes from the programming language used to produce the solution. We can see this to some extent if we look at different versions of the same program in different programming languages. Most mainstream programming languages aren't really that different from one another so the variation in complexity tends to be fairly small. Nevertheless proponents of different programming languages are always happy to highlight even marginal benefits.

Some programming languages are very different from the mainstream. For example, Zoea allows anyone to produce code simply by defining a set of test cases. It is hard to imagine a programming language that could be any simpler than this. If we compare the complexity of programs in a conventional language with equivalents in a radically different language such as Zoea then the contrast is more dramatic.

Programs in Zoea have about 50% the complexity of equivalent programs in conventional languages. Also, this figure represents a lower bound as there is still some residual complexity from the Zoea language - however small. It is not difficult to see where the complexity of conventional languages comes from. If we compare the syntax of a conventional language like Python with that of Zoea we can see that Zoea has much lower language complexity.

Copyright Zoea Ltd.

What this means is that at least half of the complexity of virtually all software is due to the choice of programming language rather than the problem that is being solved. This translates into at least half of the cost of all software development (or around $200 billion globally in 2021) that could be saved. That's quite a compelling business case for using simpler languages such as Zoea.

Learn more at **zoea.co.uk**

Copyright Zoea Ltd.